ZXZINE

Issue #2

October 2015

ZXzine

Table of Contents

Published by:
Timothy Swenson
swenson_t@sbcglobal.net
swensont@lanset.com

ZXzine is published as a service to the Sinclair ZX81 community. Writers are invited to submit articles for publication. Readers are invited to submit article ideas.

Created using Open Source Tools:

- OpenOffice
- Scribus
- Gimp
- SZ81
- EightyOne

Editorial

Here is the second issue of ZXzine. I'm not able to get stats on the Google Sites page, so I don't know how many times the last issue was downloaded. All of the feedback that I received on the issue was positive, so that was good.

This issue has a little bit of programming, some history, and some of my personal experience. I am going to try to make each issue well-rounded, with articles that should interest all. Enough of this, so here is the second issue.

Dice Roller

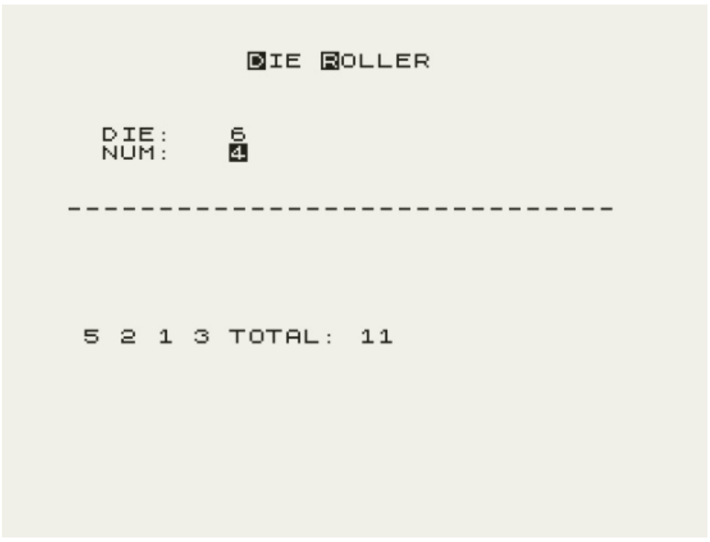
In collage I was into wargaming, both board and miniatures. Most board wargames required 6-sided dice. The miniature rule set that I was using required percentile or 20-sided dice. I had to trek to my local gaming store to pick those up. In a number of gaming magazines, I remember an ad for Dragonbone, the electronic die. It was a 6 inch long, 1.5 inches wide, plastic "wand". It had a slider switch for selecting the number of sides and a button for rolling the die. There were red LED lights with numbers next to them that signified what the die result was. The thing was design to be held in the hand and controlled with the thumb. At $24.95, it was more expensive than a lot of the wargames I was playing.

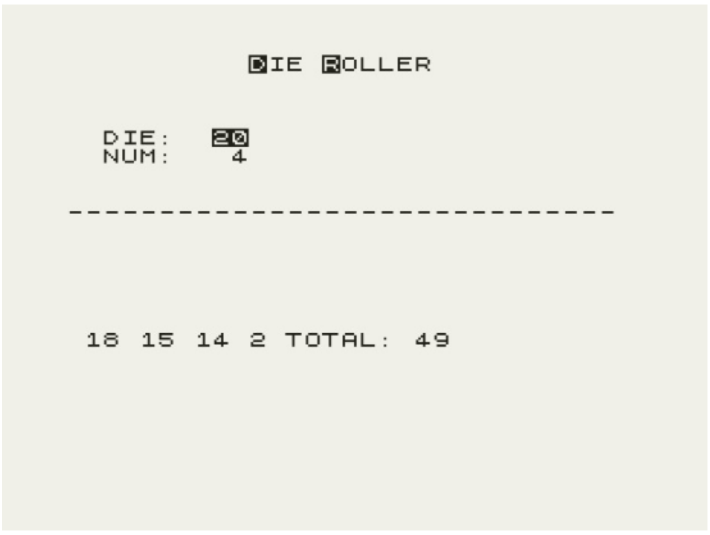
Since I never had my ZX81 out near my miniatures table, I never really tried to create my own version of Dragonbone, but many years later it seemed like an interesting exercise. I wanted the Die Roller to be as handy as the Dragonbone. No entering numbers, but using keys to select the type of die and the number. The up and down arrow keys (7 and 6 respectively) move the selection from the type of die to the number of dice. The selected item will be in inverse characters. The left and right arrow keys (5 and 8 respectively) will increase and decrease the selected value. If the number of dice is 2 and the right arrow key is hit, the number will increase to 3.

Once a selection is made on both the die and number of die, then the 0 (zero) key is hit to roll the dice. The outcome of each die roll is displayed. Since a number of gaming methodologies require a "To-Hit" roll, the dice are added up and the total is displayed. To exit the program, hit the 1 key.

The type of die are: 4-, 6-, 8-, 10-, 12-, and 20-sided. The range of number of die are from 2 - 5.

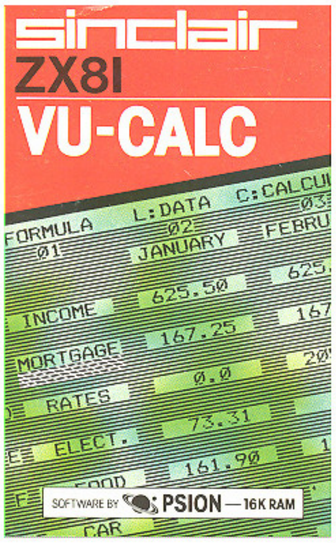
Since I was learning how to use Z88DK with the ZX81, I implemented the program in C, utilizing the different libraries like input.h, and time.h. The Z88dk documentation is not the best, but I had found examples of the routines that I needed to use.


```
              DIE ROLLER

  DIE:    6
  NUM:    4

------------------------------


 5 2 1 3 TOTAL:  11
```


```
              DIE ROLLER

  DIE:   20
  NUM:    4

------------------------------


 18 15 14 2 TOTAL:  49
```

VuCalc

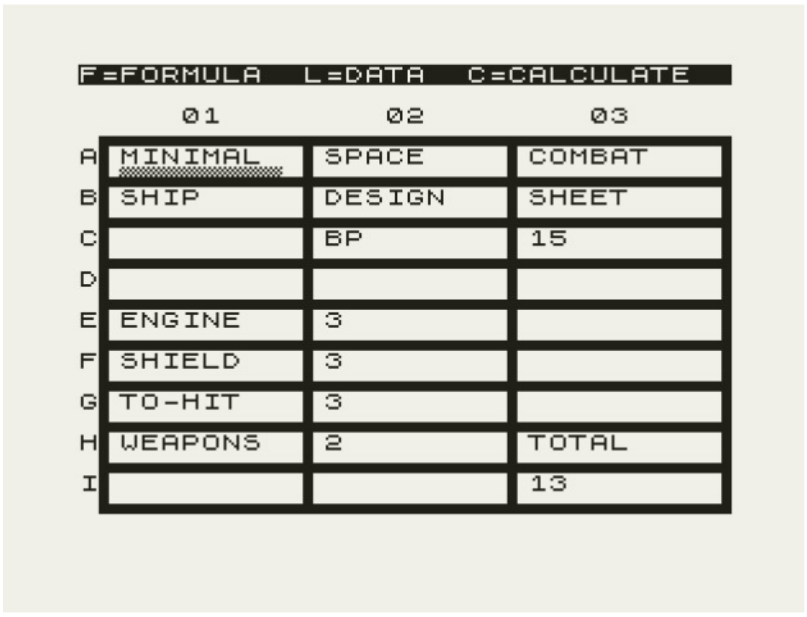
A year after getting my ZX81, I started learning about spreadsheets. I was working at a software store that was branching out into hardware and the spreadsheet was the killer application at the time. I started off learning the classic Visicalc and then moved onto Lotus 1-2-3 (version 1A). At my next job, I did almost nothing but spreadsheet work.



Original Sinclair cover for Vu-Calc

I knew Vu-Calc was available for the ZX81, but I never really tried it at out at the time as I did not have a need for a spreadsheet. Only recently did I decide to try it out and see how well it works.

When compared to other spreadsheets like VisiCalc, Vu-Calc is very limited. The calculations or formulas are very simple using just plus (+), minus (-), multiply (*) and divide (/). The formulas also allow the use of parentheses (). There are no functions, like summation, etc. It goes have the option of copying a formula and letting the calculation references be relative (changing with the copy) or static. This is helpful when doing "what-if" calculations to have a value go up a certain percent each time period, like a month).



```
F=FORMULA   L=DATA   C=CALCULATE
     01          02          03
A  MINIMAL     SPACE       COMBAT
B  SHIP        DESIGN      SHEET
C              BP          15
D
E  ENGINE      3
F  SHIELD      3
G  TO-HIT      3
H  WEAPONS     2           TOTAL
I                          13
```

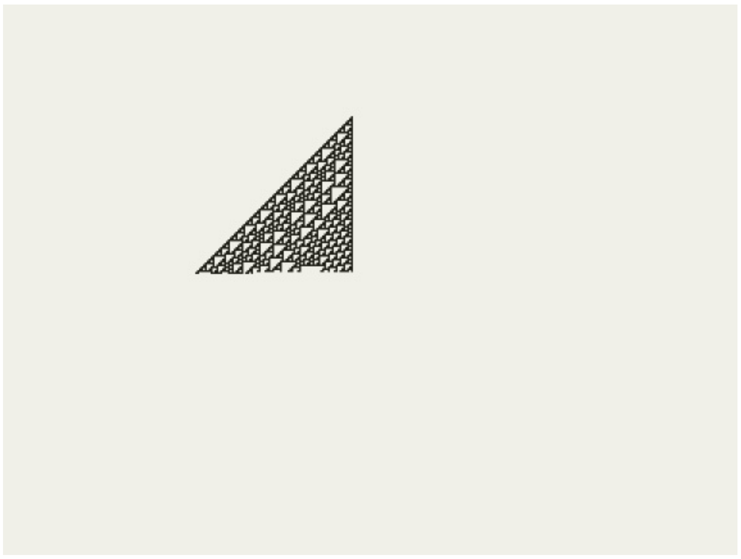
To me, the functions really make up the spreadsheet, so I was not sure if VuCalc was going to be useful. Interestingly, I did find a use for it. Some years back I designed a space combat wargame. The players are allowed to design their own ships, based on build points. Using VuCalc, the player can tweak with the different values for the ship, and VuCalc keeps a running total of the build points used. With just a few cells to add up, it was fairly simple to create the formula to do that. If it has been a few more cells, it might not have worked. The end result is a useful spreadheet.

Timex/Sinclair cover for Vu-Calc

Using HRG-ms

Matthias Swatosch has created a Hi-Res package, HRG-ms, for the ZX81, using a "graphics capable RAM pack". It is a toolkit that loads above RAMTOP and can be called via a USR call. When I first came across the package, I thought that to use it, I would have to type the program in on the ZX81 and not use a cross-over tool like zxtext2p. Since I had made the switch to Linux, I could not find a native emulator that did Hi-Res graphics, so the idea of HRG-ms was set aside.

Recently, Erik Olofsen has made changes to my favorite emulator, sz81, to support WRX hi-res graphics. I also realized packages like HRG-ms could be used with a cross-over tool like zxtext2p. Once the HRG-ms package is loaded above RAMTOP, I could load in a .P file without the emulator doing a soft reset.


Rule 110 after running for about 20 minutes

The HRG-ms package comes in two flavors, one for 16K and one for 64K. Each version provides a number of Hi-res screen banks. The 16K version provides 5 screen banks, but it only leaves you with about 6.5K usable for your program. The 64K version has 4 screen banks and leaves you with about 13K usable for Basic.

HRG-ms gives the BASIC and Assembly programmer access to a number of high resolution routines, such as plot, line, circle, box and polygon, with a total of 30 Hi-res commands. There is even a way to do User Defined Graphics. The screen resolution is basically the same as the Spectrum or T/S 2068, with 256x192 pixels. The package also sets up a number of graphic screens that can be switched between. A program listed can be listed on the non-HRG screen, the hitting the 0 and 9 keys, the HRG screen can be switched to. The package comes with an 18 page manual (both English and German) that provides a lot of detail on how the package works and how to use it.

To merge an existing BASIC program with the HRG-ms package when using sz81, you load a .P file in via the LOAD command. If you do it with the GUI, the system will do a soft reset and all memory is cleared. The LOAD command will find a .P file in the last directory it looked in. So, I put the HRG-ms .P file in the same directory as my .P files and use the GUI to load it in. Then when I want to load in the file blah.P, I just type in LOAD "BLAH" and it will find the .P file and load it.

My first test was a simple BASIC program that drew a line across the screen. I typed it in in a text editor and ran it through zxtext2p to get a .P file. I loaded in the HRG-ms package, typed LOAD, and started my program. There was a thin line drawn on the screen. It worked and that was easy.

For a more complete test of HRG-ms, I decided to use my Cellular Automata program that I have been writing for many different Sinclair computers and different languages. The program is written in BASIC without line numbers and I use zxtext2p to convert that to a .P file. I modified the ZX81 version to use the same screen resolution as the Spectrum version, which was simple to do. I then added the HRG commands to enable HRG, clear the screen bank and plot points.

With minimal changes, my program was now producing graphics that are amazing for the ZX81. There has been some discussion on the ZX81 forum on how hard HRG programming is. With the HRG-ms toolkit, converting low-res ZX81 programs is fairly simple. The problem with higher resolution is now the program has more to do. In the old ZX81 resolution it went at a moderate pace, the new resolution is very glacial. The HRG-ms package gives you better graphics, but it does give you a faster processor.

MicroAce
MICROCOMPUTER Made in USA

MicroAce

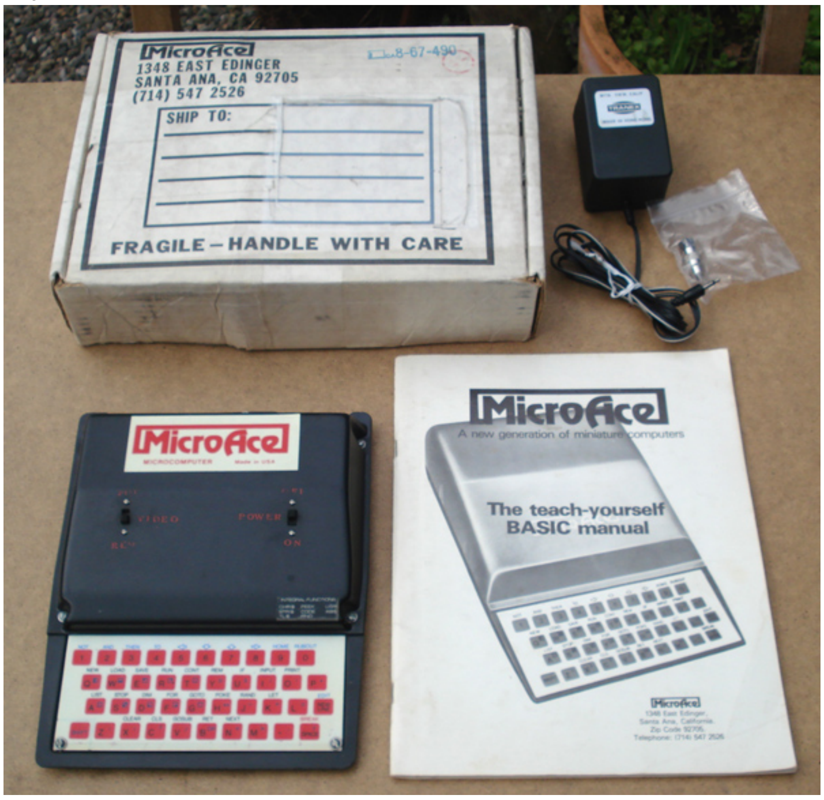
Soon after the ZX80 came out in the UK, MicroAce, an American computer company, started selling a clone of the ZX80. They used the same ROM, but made minor adjustments to the hardware. After a while, Sinclair Research Limited discovered what they were doing and took them to court. The final resolution was that MicroAce became a licensee of the ZX80 and ZX81, continuing to sell the MicroAce only in the U.S.

MicroAce was located at 1348 East Edinger, Santa Ana, California. It is unclear when the MicroAce first became available.

Before MicroAce went national, Sinclair Research Ltd. discovered the MicroAce and sued the company for copyright infringement. Exact details of the suit are hard to come by, but it seems the suit was initially thrown out as the Judge could not physically see the similarities between the two ROMS. Eventually Sinclair did win the case because of the similarity of the two keyboards. Since MicroAce copied the unique one-key entry system, the Judge could see that it was not a coincidence that they were the same. Another version of the story was this:

"The Microace designer copied the ROM from the ZX-80, but cunningly swapped two data bus lines around on the PCB layout. This resulted in a hex dump printout that was different. When Sinclair sued Microace, complaining the ROM code was a direct copy, Microace produced copies of the hexdumps (hexadecimal listings of the ROM contents) of the two ROMs. "

The results of the suit was that MicroAce licensed both the 4K and 8K ROM from Sinclair., but they were to be only sold in the United States. One user on pcmuseum.ca , said that the Comp Shop in Barnet (North London) imported the pre-built version of the MicroAce and sold them as refurbished.


MicroAce
1348 EAST EDINGER
SANTA ANA, CA 92705
(714) 547 2526
SHIP TO:
FRAGILE – HANDLE WITH CARE
MicroAce
MICROCOMPUTER Made in USA
VIDEO
POWER
ON
MicroAce
A new generation of miniature computers
The teach-yourself BASIC manual

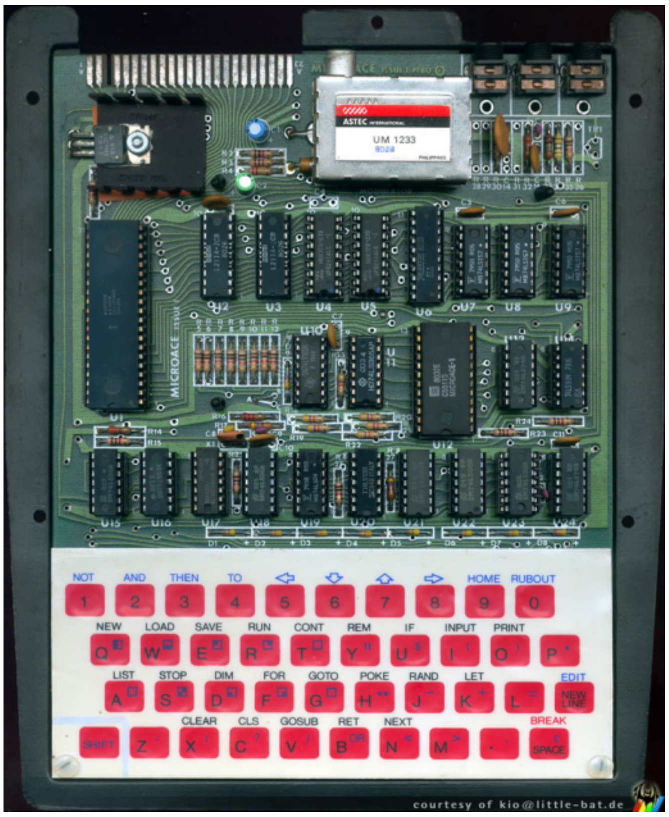
MicroAce kit package with manual and power supply

The first national advertisement was in October 1980 in Popular Electronics Magazine. The top part of the Ad did have a statement saying "licensed by Sinclair Research Ltd." The first review of the MicroAce was in the April 1981 issue of BTYE magazine.

The MicroAce was sold in kit form and came with all the parts necessary to built the computer, including cables for connection to the TV and the cassette player. The size of the MicroAce is 23.2 cm by 18.8 cm by 4.1 cm (9 1/2" deep, 7 3/8" wide, 1 5/8" high). The system was on a single PCB, with the keyboard built onto the PCB. The keyboard was similar to the ZX80, whereas the ZX80 keyboard was two layers of plastic, the MicroAce as a single layer of plastic over the PCB and when a key was depressed, it made contact with metal on the PCB. The case for the MicroAce was similar to the ZX80 case in that it used plastic pop rivets to hold the case together. The display of the MicroAce was white letters on a black background. An early reviewer found that the black was more grey, but with adjustments of the contrast and brightness, the grey could be made to look more like black.

The MicroAce was cheaper than the ZX80. The base kit was $149.00 for 1K. The 2K kit was $169.00. The manual was $10. For those who had purchased the 1K kit and wanted to upgrade to 2K, the cost was $29.00 . The ZX81 was selling for $199.95.

The manual contained both the construction details



and the BASIC user guide. A reviewer described the contraction part of the manual as very basic and relied on the customer as having experience with soldering and electronic builds. For those that had problems building the kit, MicroAce stated in their ads that "if you are unsuccessful in constructing our kit, we will repair it for a fee of $20.000, post and packing."

The BASIC part of the manual was described more as a BASIC reference guide than a tutorial for learning BASIC. The manual was short on examples of how to use the different commands.

In BYTE magazine, January 1983, review of the Timex/Sinclair 1000, the author mentioned this about the MicroAce.

The Microace company sells a modification for the ZX80 that allows a ZX80 owner to have the equivalent of a T/S 1000. Unfortunately, although the additional logic board is small and contains only seven ICs, the board won't fit inside the ZX80's case. But if you really want the continuous display, the upgrade is only $29.95 from Microace (see table 1). It works fairly well, but the board is not made by Sinclair, and I had problems with it. Microace was prompt in responding to my request for help, but its response was that I must have assembled something wrong or that something wasn't working properly. The latter turned out to be the case. After I replaced a 74LS00 chip, the modification board worked fine.

The MicroAce was mentioned in one scientific periodical, Behavior Research Methods & Instrumentation. In Volume 13, Issue 5, Jerry O'Dell, of Eastern Michigan University, submitted an articled entitled "The MicroAce: An inexpensive computer controller."

It is unclear when MicroAce finished trading. The 1983 article from BYTE puts them into early 1983.

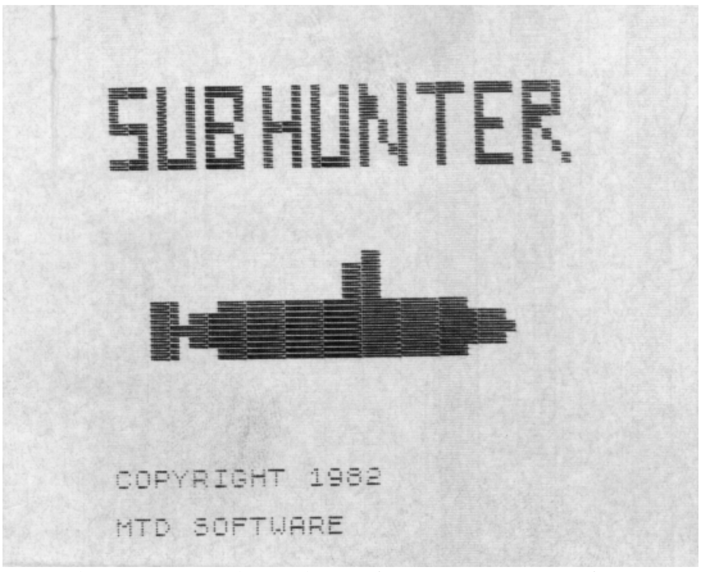
My Start with the ZX81

At the end of my Junior year of high school (spring 1981), I was taking an interest in computers. A number of home computers were out. I knew a few people that had them. On friend had an Apple II and other had a TRS-80 Model I. My friend and I were both thinking about getting one.

The day we took our SAT test, we walked from the test site to the local computer stores in downtown Hayward, California. Our first stop was the Byte Shop. We walked in and asked them what was their cheapest computer. They said "an Apple II at $1,300". After the shock wore off, we politely excused ourselves. Just a couple blocks way was Computerland. We walked in there and the cheapest computer was the Vic-20 for $300, plus $100 for the cassette unit. That was more in my price range. By the time my friend and I left, we were almost sold on the Vic-20.

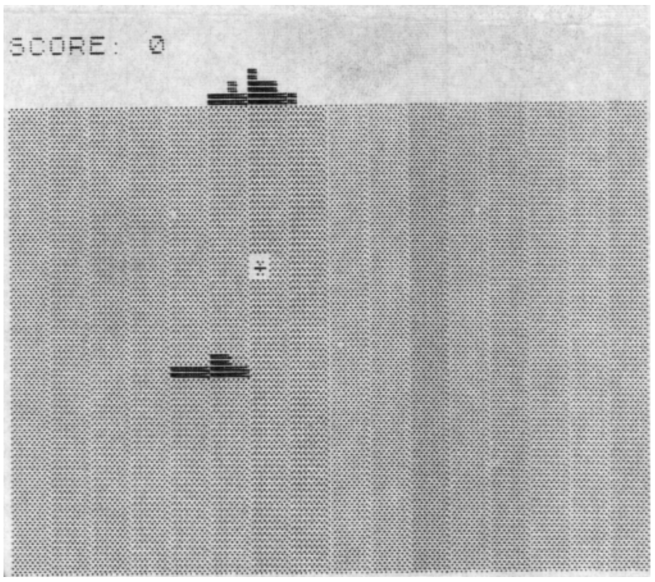
A number of months later, I was reading the October issue of Popular Science when I saw an article on this tiny little computer from England. It was only $150 and had this neat little printer to go with it. I thought it was intriguing, but did not think much about it, because it was not available in the US.

The next issue of Popular Science had a two page for the ZX81. The bright yellow background made the ad stand out. For $250, I could get a computer with 16K of memory. It had no sound, no colors and only 64x44 graphics, but it had 16K. I thought memory was more vital than color and sound. My friend bought a TRS-80 Color Computer with color, sound, and 16K for $600. Not having a checking account, I gave my Mother the $250 and she wrote a check for the computer.



Only print out that I have from an MTD Software game

The ZX81 arrived the day before Thanksgiving. I had a number of free days to set it up and experiment with it. The first thing my Step-father said when he saw the small size of the ZX81 was "You spent $250 on that?" Before I could get too far along, I needed a cassette player. A quick trip to the local mall took care of that.



When I first started with the ZX81, I was using the spare TV in the family room. I set the ZX81 on the floor, laid myself on the floor and started typing in short programs. After 10 minutes, the ZX81 would crash. It did it over and over again. Then I realized that the slots in the bottom of the ZX81 were for cooling and that conflicted with the shag carpet I was sitting it on. Once I used a better surface, all was well.

My first experiments with saving programs was horrible. I could save to tape, but I could never load. I could hear the noise on the tape, see the screen get all messy, but nothing ever loaded. I discovered that with this tape player, you had to push both Play and Record at the same time. The tape player in my stereo only need the Record button to be pushed. Suddenly, I could load from tape. I figured out that I had to run the volume all the way up until the ZX81 load screen was all black (from too much noise) and then decrease it slightly until I could see some solid horizontal bars. Once I had the volume set, I never had any problems loading.

Like other ZX81 users, I had the problem with the 16K memory pack wobble. The reason for the wobble was that link between the ZX81 and the memory pack was very tight. You could not move the ZX81 because the memory pack would shift on the external connector, losing connectivity. I figured if both the ZX81 and memory pack were attached to something stable, I could move them both together. I found some scrap peg board in the garage and cut it to be about the size of the ZX81 and memory pack. I then duct taped the ZX81 to the board, attached the memory pack and duct taped that to the board. Now when I moved the board, the ZX81 and memory pack stayed together. It was a 5 cent solution to my problem. I remember seeing all those ads that solved the rampack wobble, some for as much as $25. I wondered why people spend money on those items when a simple solution was available.

I remember somehow picking up a copy of the November/December 1981 issue of SYNC and spending hours going over each page of the magazine. Some of the first programs that I typed in (outside of the ZX81 manual)were from that magazine. I thought it was rather odd to see ads for UK based companies. In those days not many people had credit cards and the ads said that payment must be made in Sterling by International Money Order, which most Americans really had not heard of. Needless to say, I did not purchase any software until it started showing up in stores when the T/S 1000 came out.

After high school, I really started working on ZX81 programs, writing a number of games based on programs listed in "BASIC Computer Games" and "More BASIC Computer Games" by David Ahl. After having a number of games written and tested, I thought I should start a software company.

I filed the paperwork for a business license, got a home business permit, and purchased a two tape cassette deck that I could use for making copies. I called the company "MTD Software" named after myself and two friends; Mike, Tim & Doug. I tried advertising in a local free classified newspaper and had no luck. I attended a small computer show at a local high school and sold two tapes. That was the whole entirety of the sales for MTD Software. College and work took over the majority of my time.

Hidden Cave - A Fighting Fantasy Game

Back in the 1980's, I spent some of my free time playing wargames and board games. I looked into solo gaming because I wanted to be able to game when I wanted and not have to schedule time with a friend. At some gaming store I found some Warlock magazines. These were magazines for the Fighting Fantasy game books that were very popular in the UK, but did not make it over to this side of the pond.

The Fighting Fantasy rule system was very small, since most of the control of the game was done by the books. It was also similar to a number of other role playing systems like Tunnels & Trolls and The Fantasy Trip. I never did get around to playing the short adventures in the magazines, but I kept them with my other gaming stuff.

Recently I was looking for ideas for ZX81 programming and I remembered the Warlock magazines and the Fighting Fantasy system. I'm not much into adventure games, either writing or playing, but I thought it would be interesting to code a short example of such a game.

What came out of this is Hidden Cave, a short dungeon crawl adventure that uses the Fighting Fantasy combat system. I wrote the code such that it would be fairly easy to expand the game for other adventures.

At the beginning of a game, the player generates a character that has a number of Stats or characteristics like skill, stamina, and luck. These status are used for combat against monsters or "bad guys" or with traps, like arrows that shoot from holes in the wall (ala Indiana Jones).

The general style of a Fighting Fantasy game is one of reading a numbered paragraph and then taking some action at the end of the paragraph. The decision is usually something like "Turn left" or "Go through the right tunnel", or something like that. A certain points the player will encounter either monsters or traps. With monsters, the combat system comes into play. Players can be hurt or killed when fighting a monster. Traps do not require combat and only require a roll against luck.

Like all adventure games, players go through locations. In a dungeon it would be a room, or in a cave it would be a different section of the cave. The concept of rooms is used in this game.

Playing the Game

Playing the game if fairly simple and there are limited instructions in the game. Most of the Fighting Fantasy rules are coded into the game so the player does not have to worry about them, but only worry about the outcome. Hidden Cave is not a long adventure not a very exciting one, but it is a simple example to demonstrate the core part of the game.

Programming the Game

The hoped use for Hidden Cave is for others to use the "engine" to create longer adventure games. There are a number of subroutines that form the game "engine":

chargen - Subroutine for character generation. This subroutine randomly generates a character based on the FF rules. The player has the option of keeping the character or they can have the system randomly generate another one. Once the player accepts, the character stats is used throughout the game.

combat - Subroutine for monster combat. This subroutine takes the players stats and run combat against the stats of the monster. Combat has three outcomes, player kills monster, monster kills player, or player flees combat.

trap - Subroutine for handling traps. This subroutine handles traps that are put in the game to challenge the player. A random number between 1 and 6 (a roll of a die) is generated and compared with the players luck. If it is lower than luck, then the player is fine, else the player looses stamina.

roomX - Subroutine for each room. The core part of the game in the room. Each room has a description, one or more exists (unless it is an end room) and possibly a monster or trap. If there is a monster or trap, certain variables are set and either the combat or trap subroutine is called. At the end of each room is a list of where to go next and code for deciding where to go.

The hardest part of the game is creating a map of the locations of the rooms and how they are intertwined with each other (in mathematical terms, the layout of the rooms would be considered a tree graph). The text description of the rooms is the heart of the story of the adventure and takes a lot of work. Each room is a subroutine called roomX, where X is a number from 1 to X. The game should start in Room1 and move from there.

I hope the code for the game is fairly easy to read. The utility, zxtext2p, is needed to convert the code into a .P file for a ZX81 emulator.